Comp 4300 Project

Enhancing Network Efficiency : Integration of MAC protocol using dynamic bandwidth allocation AND Selective repeat with cumulative acknowledgement

Harry Pu [puh@myumanitoba.ca](mailto:puh@myumanitoba.ca)

Samuel Ning [ningyx@myumanitoba.ca](mailto:ningyx@myumanitoba.ca)

Comp 4300 Winter 2024

***Abstract*—** ***The purpose of this report is to explore the integration of various MAC protocol algorithms, including Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA), and Random Access Scheme. In this project, the main objective is to determine the benefits of combining various MAC protocols and analyze the throughput of these hybrid approaches. The modifications include combining FDMA and random access and combining FDMA with TDMA, with and without dynamic time slot adjustment. Through running network simulations using the Python simulation framework SimPy, it is concluded that these hybrid approaches do improve the overall throughput.***

***Keywords— MAC protocols, FDMA, TDMA, random access, selective repeat, network algorithm.***

# Introduction

*A. Motivation*

Selective repeat and MAC protocols play a critical role in the transport layer and data link layer in the OSI model. In this project, a few modifications to these protocols are made and simulated using the Python event-based simulation framework SimPy, aiming to improve the overall throughput of transmissions.

The selective repeat mechanism aims to provide an efficient way to ensure data integrity in the transport layer by selectively resending corrupted or lost packets. It uses selective acknowledgment and retransmission to maintain data integrity even in poor network conditions.

MAC protocols, on the other hand, operate in the data link layer. Their primary purpose is to define how devices share the same medium effectively. FDMA allows multiple devices to transmit at the same time by dividing the channel into different frequencies. However, the fixed allocation can result in unused bandwidth and lead to lower efficiency. TDMA coordinates multiple devices by allocating time slots to different devices in a cyclic manner. But TDMA also suffers from wasted resources due to inactive devices holding the bandwidth during their allocated time slots. Although the random access scheme does not suffer from this problem, an increase in the number of devices can result in a higher number of collisions and inefficiency. To address these problems, a mechanism that allocates bandwidth dynamically to different devices based on each device's bandwidth requirement is needed to effectively allocate the shared channel's bandwidth and improve overall throughput.

*B. Real-Life Applications*

When considering the benefits of using a dynamic approach to allocate bandwidth in the MAC protocol, it is important to consider ways to determine the bandwidth requirements of different devices. Some approaches include:

* Using statistical approach to determine devices bandwidth usage, dynamically allocate bandwidth resources to devices based on device's historical usage in different time period of the day.
* Utilizing RNN machine learning model to process each device's short term traffic data and make prediction of their bandwidth usage.
* Using AI model to analyze device's usage pattern and allocate bandwidth resources.
* Supporting device's bandwidth allocation based on their priority.

Next this report will discuss MAC protocols simulation set up and Analysis of the results.

# II. MAC PROTOCOLS SIMULATION

**To Run:** All MAC Protocol simulations use the python simpy simulation Framework. To run each simulation, ensure simpy is installed (**pip3 install simpy**). Then run each python script: **python3 script\_name.py.**

**Simulation Conditions and Assumptions:** All simulations simulate 9 nodes sending data with various loads from 1 to 9, where node 1 has the lowest data load and node 9 has the highest. It is also assumed that the time taken for 1 node to send 1 packet under full bandwidth is 2 time units. The total bandwidth of the channel does not change. All simulations are run for 1000 time units, and their performance metrics are shown in a graph.

*A. FDMA: Uniform vs. Stratified Bandwidth allocation (FDM.py)*

Figure 1.1 shows the number of frames sent by the 9 nodes with various loads (10 to 90), but with the same bandwidth. Figure 2.1 shows the frames sent by the 9 nodes that are allocated bandwidth speeds proportional to their data loads. That is, a node with a higher data load is assigned to a band with higher bandwidth, and vice versa (see mechanism in Firugre 1.2 and 2.2). It is clear that stratified bandwidth allocation based on the node's data load results in a higher overall throughput of 0.458 packets/time unit.

A graph with colorful lines

Description automatically generatedA table with arrows and numbers

Description automatically generated with medium confidence

Figure 1.2 FDMA with uniform bandwidth mechanism.

Figure 1.1. FDMA with Uniform bandwidth allocation

A graph of different colored lines

Description automatically generatedA diagram of a band

Description automatically generated

Figure 2.1. FDMA with Stratified bandwidth allocation. Figure 2.2 FDMA with stratified bandwith mechanism.

*B. FDMA+ TDMA: Reduce idling time slot in TDMA by intergrating TDMA into different bands.(TDM.py, FDM\_TDM.py)*

Figure 3.1 shows the packets sent by using TDMA. All 9 nodes share the same bandwidth in a single channel, resulting in an overall throughput of 0.193 packets/time unit and 193 packets sent (see mechanism in figure 3.2). Then, the 9 nodes are divided into 3 groups based on data load: low (10,20,30), medium (40,50,60), and high (70,80,90), and placed into 3 bands with low, medium, and high bandwidth speeds, respectively (see mechnism in figure 4.5). Figure 4.1 shows that this approach increases the throughput by around 39%, from 0.193 packets/time unit to 0.268 packets/time unit. Figures 4.2, 4.3, and 4.4 show the packets sent by each band, with increasing throughput from band 1 to band 3. This is expected since band 1 has the lowest bandwidth and nodes with lower data load, while band 3 has the highest bandwidth and nodes with high data load.

A graph of a graph

Description automatically generated with medium confidence

Figure 3.1. Regular TDMA

A table with numbers and text

Description automatically generated with medium confidence

Figure 3.2. Regular TDMA Mechnism

A graph of a number of data

Description automatically generated with medium confidence

*Figure 4.1. FDMA+TDMA. All 3 bands*

.

A graph showing a line of data

Description automatically generated with medium confidenceA graph of a graph

Description automatically generated with medium confidence

Figure 4.2 FDMA+TDMA. Band 1 (low bandwidth) Figure 4.3 FDMA+TDMA. Band 2 (medium bandwidth)

A graph of a number of data

Description automatically generated with medium confidenceA table with numbers and letters

Description automatically generated

Figure 4.4 FDMA+TDMA. Band 3 (high bandwidth). Figure 4.5 FDMA+TDMA Mechanism.

*C. FDMA+ TDMA: Reduce idling time slot by dynamicly allocating time slot to nodes based on their data load within each band from B (FDM\_TDM\_DynamicTimeSlot.py)*

This simulation is an enhancement of B (see Figure 5.5 for mechnism). After grouping nodes and putting them inside a band with corresponding bandwith based on the node's data load, furthur optimization within the band can be done. In simulation B, although the overall throughput improves after combining FDMA and TDMA, Figure 4.2, 4.3 and 4.4 shows that the nodes within each band still transmitt at similar rate as the lines represending nodes in each bands cluster together. Thus in this simulation each node is assigned with dynamic number of time slots. For instace, in band 1, node 1 gets 1 time slot to trasmitt, while node 2 and node 3 with higher data load get 2 and 3 time slots. In band 2, node 1 gets 1 time slot to transmit, while node 2 and node 3 get 2 time slots each.

The results of this simulation can be found in Figure 5.1: the overall transmitted packets total 466 and the throughput is 0.268 packets/time unit, which is 74% higher than that in the FDMA+TDMA simulation without dynamic slot allocation, and 140% higher than in the TDMA simulation. Comparing Figures 4.2, 4.3, and 4.4 to Figures 5.2, 5.3, and 5.4, it is clear that the nodes within a band start to differentiate in the number of packets transmitted. The nodes with lower data loads (node 1 in each band) do not lose too much throughput, while the nodes with higher data loads (nodes 2 and 3 in each band) start to pick up speed when utilizing dynamic time slot allocation within each band.

A graph of a number of packets

Description automatically generated with medium confidence

Figure 5.1. FDM+TDM with dynamic time slow allocation. All 3 bands

A graph of a graph

Description automatically generated with medium confidenceA graph of a number of different colored lines

Description automatically generated

Figure 5.2 FDM+TDM with dynamic time slot allocation. Band 1 Figure 5.3 FDM+TDM with dynamic time slot allocation. Band 2

A graph of a number of data

Description automatically generatedA table with numbers and letters

Description automatically generated with medium confidence

Figure 5.4 FDM+TDM with dynamic time slot allocation. Band 3 Figure 5.5 FDM+TDM with dynamic time slot allocation.Mechnism

*D. FDMA+Random Access: Reducing Collision Rate in MAC Random Access by integrating random access scheme within seperate band (randomAccess.py, FDM\_randomAccess.py)*

Figure 6.1 shows the simulation result of running random access for nodes 1 to 9 within a single band. The collision rate is 45.82%. Figure 6.2 shows the collision versus frames sent for each node. Figure 7.1 shows the simulation result of integrating random access into FDMA. Nodes are grouped together and assigned to a band based on their data load. Band 1 contains nodes with lower data loads and has the lowest bandwidth. Band 3 contains nodes with high data loads and has the highest bandwidth (See figure 6.3 and figure 7.3 for mechnisms).

Comparing this hybrid approach to the original random access protocol, the collision rate drops by almost half, from 45.82% to 22.74%. Figure 7.2 shows the collisions versus frames sent for this hybrid approach as well. Compared to Figure 6.2, where most nodes have a higher number of collisions than packets sent, the hybrid approach effectively reduces collisions in all nodes. However, node 3 in band 2 and nodes 2 and 3 in band 3 seem to be hogging all the bandwidth due to inaccuracies in the SimPy framework simulation. This is to be improved in the future.

Overall, combining FDMA with Random Access isolates the nodes in each band, effectively reducing the collisions and improving the overall throughput by about 16.5%, from 0.38 to 0.46 packets/time unit.

*.* A graph with numbers and a chart

Description automatically generated

Figure 6.1 Random Access Transmissions

A graph with red and blue bars

Description automatically generatedA person pointing at a white square

Description automatically generated

Figure 6.2. Random Access Collisions Figure 6.3. Random Access Mechnism(single band)

A graph with numbers and lines

Description automatically generated with medium confidence

Figure 7.1.FDMA+Random Access Transmission

A graph with blue and red squares

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 7.2. FDMA+Random Access Collisions Figure 7.3. FDMA+Random Access Mechnism

*E. MAC protocols conclusion:*

By identifying device's data load and dynamicly allocate bandwidths to them, the MAC protocols can achieve a higher efficiency. The overall transmission throughput can be improved by the following approach as shown in the simulations:

* Stratified allocaion of bandwidth based on node's load rate in FDMA.
* Reduce idling time slot in TDMA by intergrating TDMA into different bands.
* Reduce idling time slot by dynamicly allocating time slot to nodes based on their data load within each band.
* Reducing Collision Rate in MAC Random Access by integrating random access scheme within seperate band.

# III. SELECTIVE REPEAT WITH CUMULATIVE ACKNOWLEDGEMENT

In the Selective Repeat Protocol, both the sender and receiver maintain a window size, and each frame is marked with a sequence number, enabling both the sender and receiver to keep track of the frame order. The sender's responsibility is to send the frames within its window and set a time out for the frames sent. If a frame is timed out (acknowledgment not received), the sender assumes that the frame was lost in transmission, and the frame is resent. Likewise, if the confirmation frames are out of order they are buffered until they can be processed in the correct sequence. On the receiver side, the sender's frames are received, checked, and reconstructed correctly. Thus, if frames are being received is out of order, then they are stored until all preceding frames are received.

Selective Repeat with Cumulative Acknowledgment extends the Selective Repeat protocol by introducing a cumulative acknowledgment mechanism. Instead of sending acknowledgment for each frame, the receiver's response would be an acknowledgement containing a list of sequence number representing all frames received within the receiver's window size.

**Methodology:**

* **Test Scenarios:** Four test scenarios were devised, each with different frame loss rates: high (30%), moderate (15%), and low (5%).
* **Parameters:** The parameters for each scenario were kept constant, including the total frames to send (500), window size (5), timeout (1 second), and network delays (minimum: 0.01 seconds, maximum: 0.05 seconds).
* **Metrics:** Performance metrics such as throughput, efficiency, and average number of retransmissions per frame were measured to evaluate the effectiveness of Selective Repeat and Selective Repeat with cumulative acknowledgment in each scenario.

**Results:**

1. **Scenario 1: High Frame Loss Rate (30%)**

**Selective Repeat:**

To run the python script: **python3 main.py -lr 30**

A computer screen with numbers and symbols

Description automatically generated

A graph showing a number of blue lines

Description automatically generated

**Selective Repeat with Cumulative Acknowledgement**

To run the python script: **python3 main.py -cum-ack -lr 30**

A screenshot of a computer

Description automatically generated

A blue lines with white text

Description automatically generated

1. **Scenario 2: Moderate Frame Loss Rate (15%)**

**Selective Repeat**

To run the python script: **python3 main.py -lr 15**

A computer screen shot of numbers and frames

Description automatically generated

A graph with blue lines

Description automatically generated**Selective Repeat with Cumulative Acknowledgement**

To run the python script: **python3 main.py -cum-ack -lr 15**

A computer screen with numbers and percentages

Description automatically generated

A graph with blue lines

Description automatically generated

1. **Scenario 3: Scenario 3: Low Frame Loss Rate (5%)**

**Selective Repeat**

To run the python script: **python3 main.py -lr 5**

A computer screen shot of numbers and letters

Description automatically generated

A graph with blue lines

Description automatically generated

**Selective Repeat with Cumulative Acknowledgement**

To run the python script: **python3 main.py -cum-ack -lr 5**

A screenshot of a computer screen

Description automatically generated

A graph with blue lines

Description automatically generated

**Analysis:**

Selective Repeat with cumulative acknowledgment consistently outperforms traditional Selective Repeat across all scenarios. It achieves higher throughput by acknowledging multiple frames with a single acknowledgment, allowing the sender to quickly shift the window, thus enabling it to send more frames leading to increased data transmission rates Additionally, it exhibits higher efficiency by reducing retransmissions and lowering the average number of retransmissions per frame. This occurs because it acknowledges the sender more quickly, reducing the duplicate frames sent by the sender. Overall, Selective Repeat with cumulative acknowledgment's ability to acknowledge multiple frames simultaneously enhances throughput, efficiency, and reliability compared to Selective Repeat.